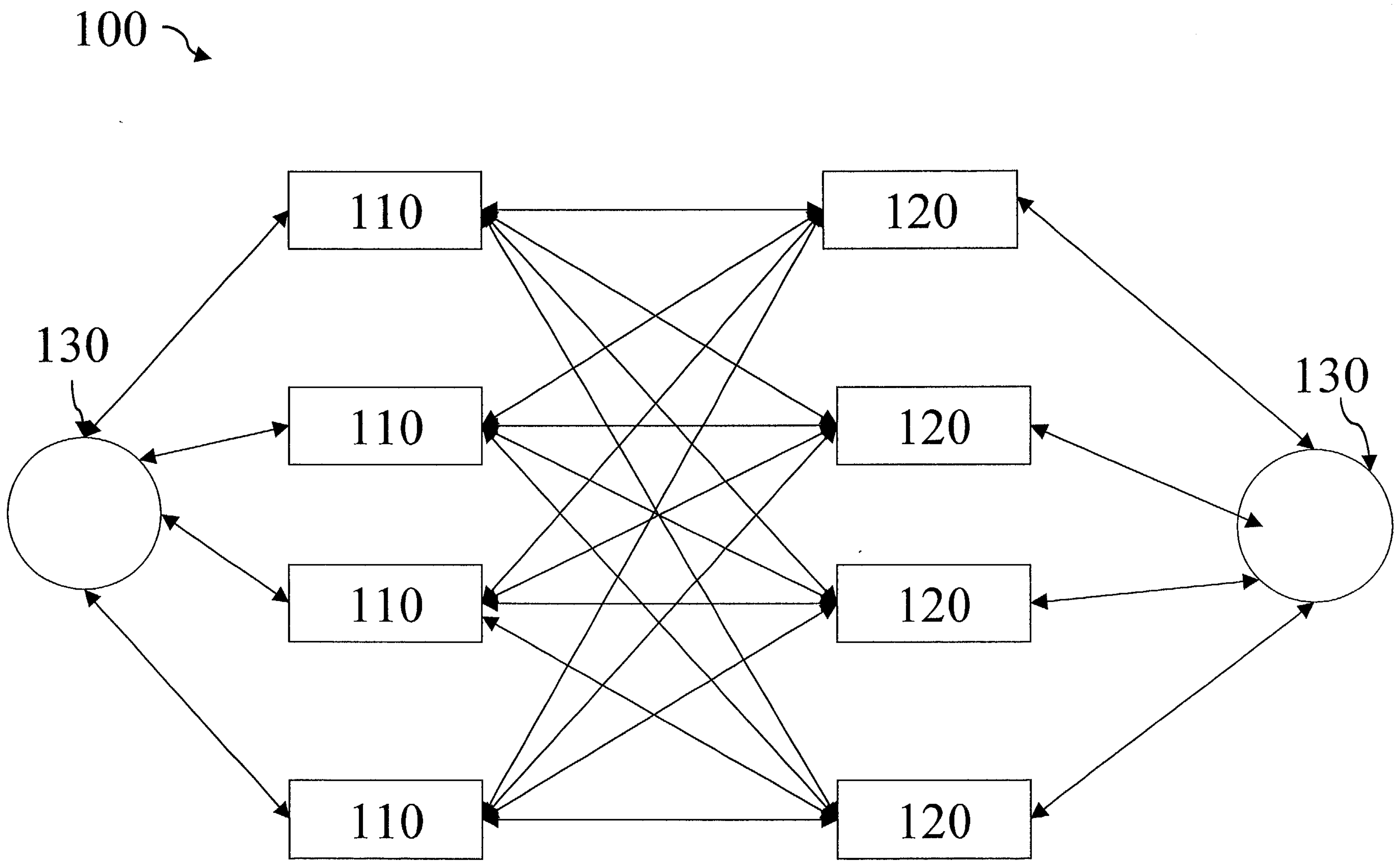

FIGURE 1

```
contend():
s ← 0;
repeat [
    $Q_1 \leftarrow \emptyset$;
    $\|_{u \in U}$ [ $status_u \leftarrow u$,try();
        $Q_1 \leftarrow \{u\} \cup Q_1$;
    ] until $(\exists Q \in \mathcal{Q} : Q \subseteq Q_1)$;
    if $(|\{u : status_u = \mathrm{LOCKED}\}| \leq b)$
        return;
    else
        $s \leftarrow s + 1$;
        $d \leftarrow_R [(\Delta+4\delta)] \ldots 2^s(\Delta+4\delta)$;
        sleep($d$);
    ] until (false);
```

FIGURE 2A: MUTUAL EXCLUSION PROTOCOL CLIENT PROGRAM

```
try():
    if (clock() - lastGranted > $\Delta + 2\delta$)
        lastGranted $\leftarrow$ clock();
        return FREE;
    else
        return LOCKED;
```

FIGURE 2B: MUTUAL EXCLUSION PROTOCOL SERVER PROGRAM

```
1)  submit(o):
2)  waiting ← true;
3)  ‖ $V_1$ ← $\emptyset$; $Q_1$ $\leftarrow_R$ $\mathcal{Q}$;
4)     $\|_{u\in Q_1}$ [$p_u$ ← u.submit(o);$V_1$←
       {$p_u$}∪ $V_1$;
5)     ] until ($\exists p$:|{$p_u \in V_1$
:p=$p_u$}|≥b+1;
6)     waiting ← false;
7)     return p:| {$p_u \in V_1$:p=$p_u$}|
       ≥b+1
8)  ‖ repeat [
9)        contend();
10)       $Q_2$ ← $\emptyset$;
11)       $\|_{u\in U}$ [⟨$\sigma_u^c$, $\sigma_u^{pc}$,
    proposer$_u$,, pending$_u$⟩←u.get(r);
12)            $Q_2$←{u} ∪ $Q_2$;
13)       ]until ($\exists Q \in \mathcal{Q}$:Q ⊆ $Q_2$);
14)       $\sum^c$←{σ′:|{u:σ′=$\sigma_u^c$  }|≥b+1};
15)       $\sigma^c$←:σ:σ.version=max$_{\sigma'\in\sum c}$
          {σ′.version};
16)       $\sigma^{pc}$←choose({⟨$\sigma_u^{pc}$,
    proposer$_u$⟩:σ  v≠⊥});
17)     completed←max
        {completed,max{v:|{u:$\sigma_u^{pc}$
        .version>v|≥b+1}};
18)     if $\sigma^c$≠⊥ ∧ $\sigma^c$.version>
        completed)
19)       σ←$\sigma^c$
20)     else if ($\sigma^{pc}$≠⊥ ∧ $\sigma^{pc}$.
    version > completed)
21)       σ←$\sigma^{pc}$;
22)     else
23)       pending←{o:∈{u:o ∈
          pending$_u$}|≥b+1};
24)       σ ← apply  (pending
  $\sigma^c$);
25)       $Q_2$ ← $\emptyset$;
26)       $\|_{u\in U}$ [u.propose(σ′ r);$Q_2$←
        {u}∪$Q_2$;
27)       ]until ($\exists Q\in\mathcal{Q}$:Q⊆$Q_2$;

28)       $Q_2$ ← $\emptyset$;

29)     $\|_{u\in U}$ [u.commit(σ,r);$Q_2$ ←
        {u}∪$Q_2$;
30)     ]until($\exists Q\in\mathcal{Q}$:Q⊆$Q_2$);

31)       completed ← max
 {completed,σ.version};

32)  ] until (waiting =
false);

33) choose ({⟨$\sigma_u^{pc}$,proposer$_u$⟩}u):
34)    S[1,2,...]←{⟨$\sigma_u^{pc}$ ,
 proposer$_u$⟩}$_u$ sorted in
descending order by ⟨σ,
proposer⟩ > ⟨σ′,
proposer′⟩⇔(σ.version >
σ′.version ∨
(σ.version =
σ′.version ∧ proposer    >
proposer′));
35)    i ← 1; count ←
[0,0,...];
36)    repeat
       [⟨σ,proposer⟩←S[i];
37)       count[σ]←count[σ]+1;
38)       i ← i + 1;
39)     ] until($\exists\sigma$:count[σ]
       ≥b+1 ∨ i > |S|)′
40)     if ($\exists\sigma$:count[σ]≥b+1
41)       return σ:count
          [σ]≥b+1;
42)     else
43)       return ⊥;
44) apply (pending, σ′):
45)  repeat [ o $\leftarrow_R$ pending;
46)     pending←pending\{o};
47)     if(σ′.reflects
        (o)=false)
48)         σ′.doOp(o);
49)    ] until (pending=∅)′
50)    σ′.version←σ′.
       version+1′
51)    return σ′;
```

FIGURE 3A: CLIENT SIDE OF AN ORDERING PROTOCOL

```
1)  submit (o):
2)      pending←pending ∪{o};
3)      sleep until
    (response(o)≠⊥);
4)      return response (o);
```

```
5)  get(r):
6)      if ® > maxRank)
7)      maxRank ← r;
8)      return ⟨¤Ł¤σ^c, σ^pc,
        prosper, pending⟩
9)  else
10)      throw RankException;
```

```
11)  propose (σ,r):
12)      if ® ≥ maxRank)
13)        maxRank ← r;
14)        proposer ← r;
15)        σ^pc ← σ;
16)        return;
17)  else
18)      throw RankException;
```

```
19) commit (σ,r):
20)  if ® ≥ maxRank)
21)    maxRank ← r;
22)    σ^c, σ;
23)    pending ← pending \
       {o:σ.reflects(o)=true};
24)    response ← response|σ.
       response;
25)    return;
26)  else
27)    throw RankException;
```

FIGURE 3B: SERVER SIDE OF AN ORDERING PROTOCOL